

The ND Emulator project

Version: 21. March 2019

Carl-Victor Sundling
Halden, Norway

sundling@getmail.no

ND Emulator User Manual

1	INTRODUCTION	6
2	CHANGES FROM PREVIOUS VERSION	7
2.1	Getting Started/Installation.....	7
2.2	Terminal emulation.....	9
2.3	Mandatory to know	10
2.3.1	BG-programs and RT-programs.....	10
2.3.2	The file system	10
2.3.3	File indexing “NO SUCH PAGE”	10
2.3.4	Terminal break strategy.....	11
2.3.5	Parity	11
2.3.6	Users.....	11
2.3.7	2-bank programs.....	12
2.3.8	Reentrant subsystems	12
2.3.9	Floating point	12
2.3.10	Misc.....	12
2.3.11	Instructions not emulated	12
3	THE EMULATOR COMMAND INTERFACE	12
3.1	upper-case <on/off>.....	13
3.2	log-in-user <user name>	13
3.3	list-users.....	13
3.4	trace-execution <on/off>	13
3.5	trace-monitor-calls <on/off>.....	2
3.6	trace-stack <address>	2
3.7	begin-trace-execution-when-at <address>	2
3.8	define-trace-area <trace no.> <lower> <upper>	2
3.9	reset-trace-area <trace number>	2
3.10	open-trace-file <output file>	2
3.11	close-trace-file	2
3.12	set-break-point <address>	2
3.13	clear-breakpoint	3

3.14	continue-after-breakpoint	3
3.15	define-histogram <start-address> <increment>	3
3.16	list-histogram [optional output file]	3
3.17	guard <location>	3
3.18	report-memory-reference <location>	4
3.19	reset-report-memory-reference	4
3.20	look-at-memory <page table> <address>	4
3.21	find-value-in-memory <value> <page table>	5
3.22	find-nonzero-in-page-table <page table no>	5
3.23	fill-memory <value> <from> <to>	5
3.24	copy-memory <from> <bytecount> <ntimes>	5
3.25	set-register-value <register> <value>	5
3.26	list-register-values	5
3.27	disassemble-memory <output file> <from addr> <to addr>	6
3.28	status	6
3.29	dummy	6
3.30	! <image file>	6
3.31	load-image-file <image file>	7
3.32	load-byte-swapped-image-file <image file>	7
3.33	set-memory-limits <lower> <upper>	7
3.34	set-2bank-mode <on/off>	7
3.35	activate-alternative-page-table <APT no.>	8
3.36	save-image-to-file <image file> <SA> <RA>	8
3.37	dump-memory <prog.file> <start address> <restart address>	8
3.38	dummy2	8
3.39	run <start-address>	8
3.40	continue-at-restart-address	9

3.41	help <command> <output file>	9
3.42	open-scratch-file	9
3.43	check-files	9
3.44	use-7bit-on-symb-files-write <on/off>	10
3.45	float <format>	10
3.46	compare-images <image file> <image file> [report file]	10
3.47	add-parity <input file> <output file>	10
3.48	remove-parity <input file> <output file>	10
3.49	byte-swap <inp.file> <outp.file>	11
3.50	od <format> <input nfile>	11
3.51	octal-to-decimal <number>	11
3.52	decimal-to-octal <number>	11
3.53	float-test <val1> <val2>	11
3.54	mode <input file> <output file>	12
3.55	exit.....	13
3.56	quit.....	13
3.57	debug <level(0=off)>	13
3.58	zz	13
3.59	assemble-to-memory <start address>.....	14
3.60	READ-BINARY <input file>	14
3.61	WRITE-BINARY <page table> <output file>.....	14
3.62	define-segment <segment number > <name>	14
3.63	load-segment <segment number> <progfile>	14
3.64	list-segments	14
3.65	delete-segment.....	15
3.66	list-SINTRAN-commands.....	15
3.67	SINTRAN-III commands description	15

3.67.1	@set-terminal-type <n>	15
3.67.2	@list-open-files	15
3.67.3	@close-file <file number (-1=all)>	16
3.67.4	@list-file <file name>	16
3.67.5	@new <user name>	16
3.67.6	@who	16
3.67.7	@place-binary <.BPUN file>	17
3.67.8	@place-image-file <.PROG file>	18
3.67.9	@go <address>	18
3.67.10	@recover <prog file>	19
3.67.11	@delete-file <file name>	19
3.67.12	@create-file <file name>	19
3.67.13	@copy-file <to file> <from file>	19
3.68	SINTRAN-info	20
3.69	find-in-file <file> <w1> <w2>	20
3.70	BCD-test <value 1> <operation> <value 2>	20
3.71	test-input	20
3.72	CPU-type <cpu type no.> <instruction set no.>	21
3.73	set-parity-when-reading-SYMB-files <on/off>	21
3.74	check-monitor-call <number>	21
3.75	silent-mode <on/off>	21
3.76	more	21
4	STOPPING A RUNNING ND APPLICATION	21
5	PLANNED DEVELOPMENT	22
5.1	Reentrant subsystems	22
6	MONITOR CALLS EMULATED	22
6.1	Nonconformance monitor calls	22
7	LIST OF EMULATOR COMMANDS	22
7.1	The PREP2B program	24
8	SYSTEM DEVICES	25
9	SEGMENT OVERLAY	26

1 INTRODUCTION

The ND Emulator can be used freely for non-commercial purposes and may be redistributed provided the receiver is made aware of this. The user manual can be copied without any restrictions.

The ND Emulator has been developed in the 'C' programming language partly using Linux and partly using Windows. The Windows version is running as a 32 bit application without a graphical user interface and is controlled via an interactive user interface in a DOS window. Users of SINTRAN will recognize the command interface with abbreviation of the commands and prompts for missing parameters.

The aim of this emulator project was to emulate the instruction set to see how fast an emulator could execute compared to a ND computer. The next step was to be able to run ND programs like the MAC assembler, an early version of the Fortran compiler (FTN) and QED. To be able to do this it was necessary to also emulate many SINTRAN III monitor calls and make an interface to the Windows and Linux file systems.

Platform

The emulator is developed using Visual Studio running under Windows 10. The emulator should run on Windows XT and newer.

Linux is currently not supported.

Multiusers.

You may start several instances of the emulator and each instance will open its private SCRATCHxx file.

A maximum of 8 simultaneous users is allowed.

The emulator has been tested using the following ND programs:

- FTN
- FORTTRAN-100-G02
- MAC
- FMAC
- QED
- NRL
- BRF-LINKER
- LOOK-FILE
- NPL
- LIST-ALL-FILES
- PED

Though some testing has been carried out with these programs the author does not guarantee that the emulator is bug-free. If you get any problems or find errors please do not hesitate to report it to this mail-address:

sundling@getmail.no

Source file editing

Ped is working, but it may be more efficient to use e.g. Notepad to edit source. It is more convenient to mark text, set the cursor to a position etc. (which is not supported by PED). Some special keyboard keys from Tandberg Terminals is not available in the emulator.

Most of the ND-programs check for even input parity which is not generated by Notepad. To cope with this issue the command:

<>set-parity-when-reading-SYMB-files <on/off>

MAC, NPL, FORT-100 etc. will now read files generated by Notepad.

Note: QED can read files without parity using MPI(0)
QED can write files without parity using MPO(0)

2 Changes from previous version

Version 1 2017

The TDV-2215 terminal is now emulated (not completely) and PED can be used to edit text files by using terminal type 52..

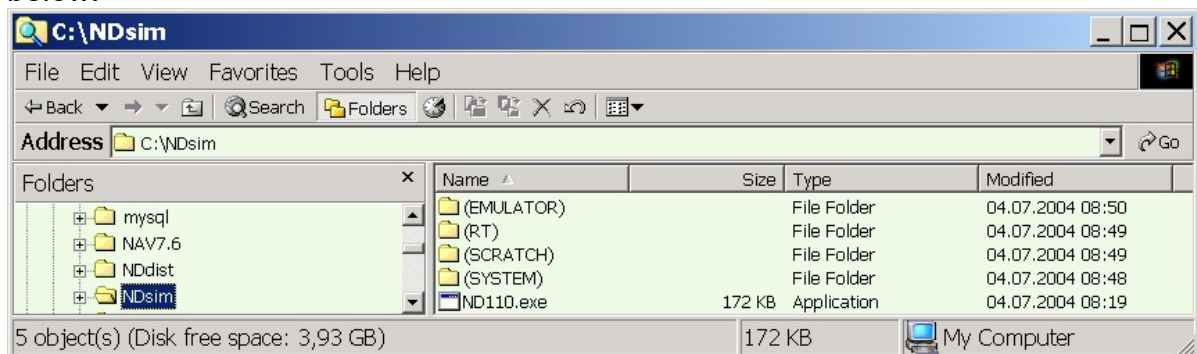
Several monitor calls have been introduced. Overlay segments function is not tested very intensive...

A number of bugs are fixed

2.1 Getting Started/Installation

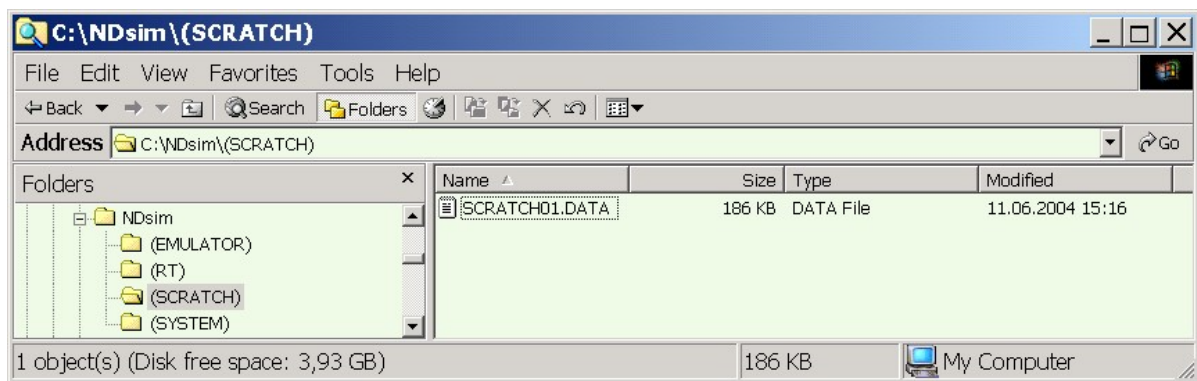
NOTE: All folder names and file names (except Ndsim and ND110.exe) should have upper case letters only!

Create a folder called Ndsim. In this folder create the folders as seen in the picture below:

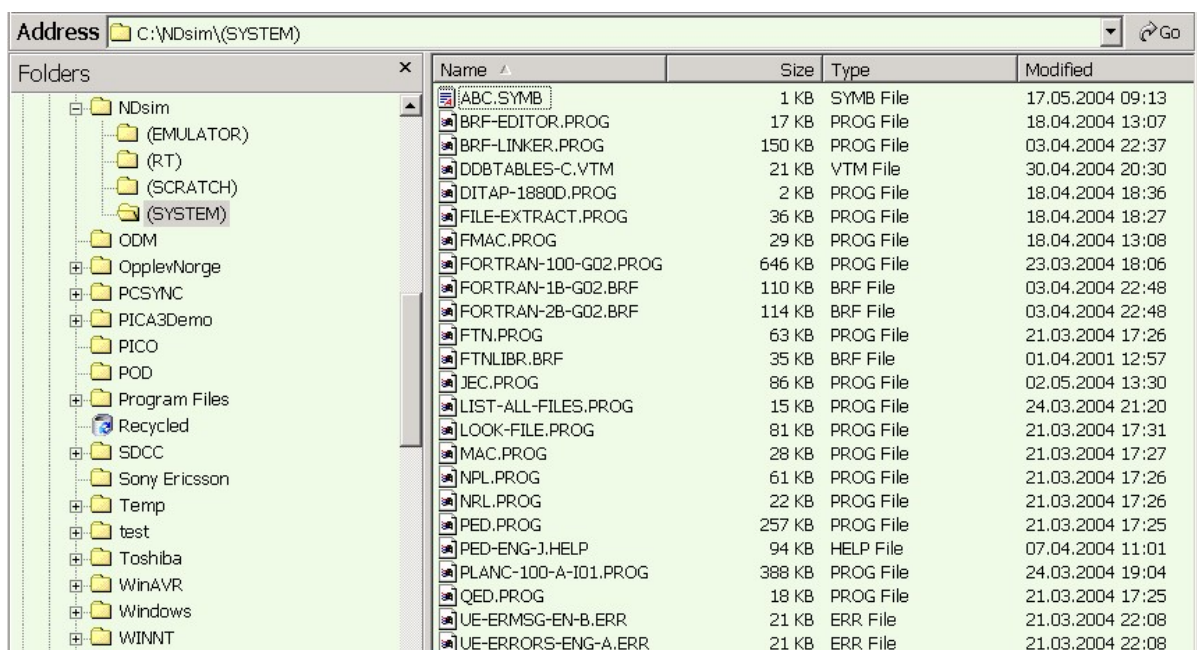


Put the ND110.exe file in the Ndsim folder.

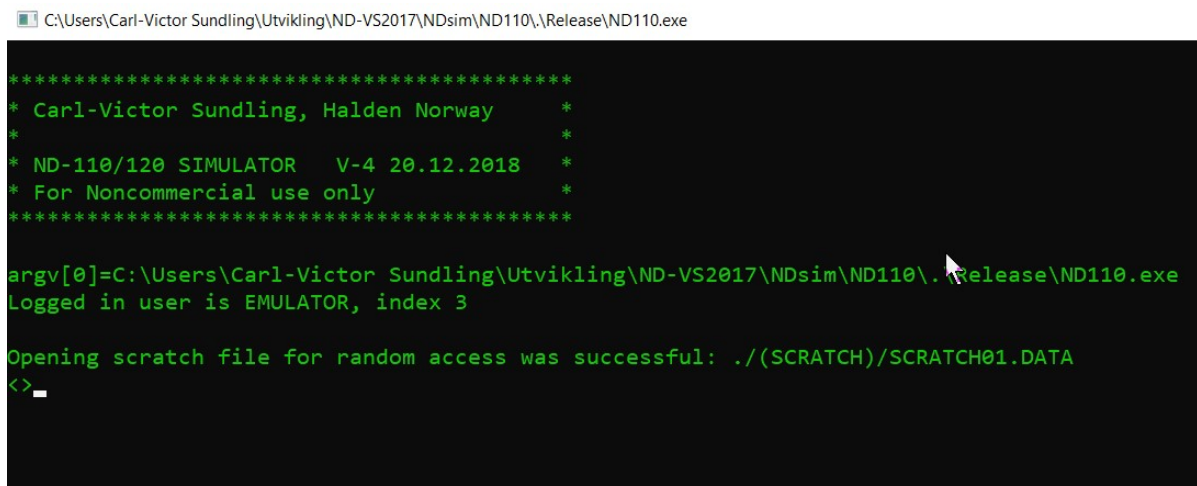
Create the file SCRATCH01.DATA in the folder (SCRATCH) as seen in the picture below:



Install the ND executables and their system files (like PED-ENG-J.HELP) in the (SYSTEM) folder.



Create a shortcut to the ND110.exe file on your desktop and double click it to start the emulator. The emulator is started in a DOS window and you are automatically logged in as user EMULATOR. Typing QED should now start the QED editor



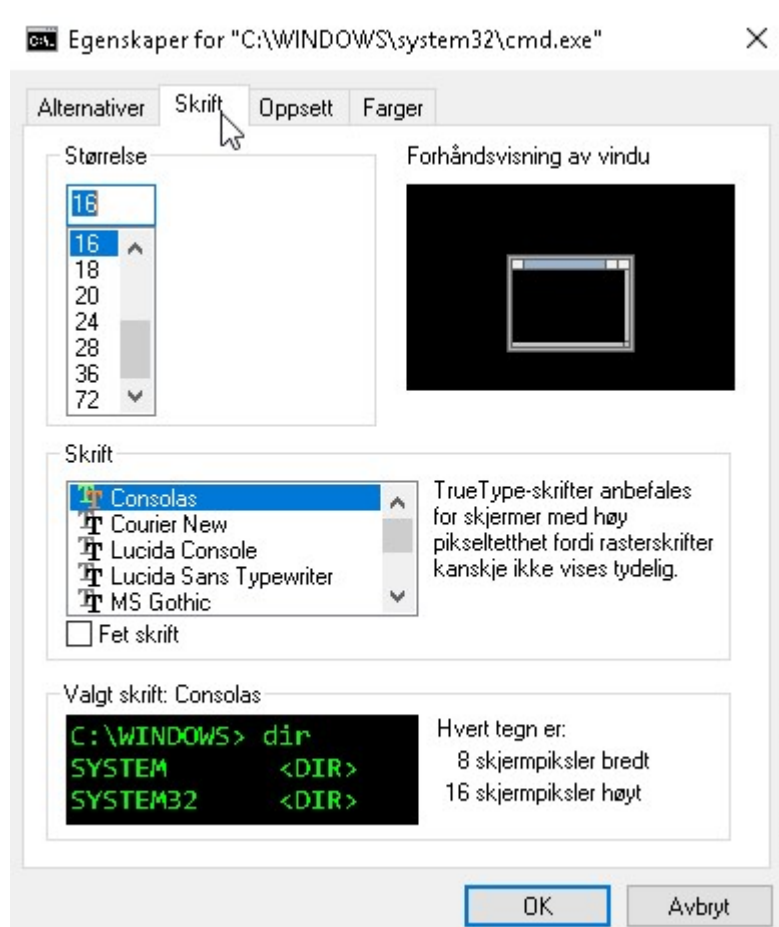
2.2 Terminal emulation

The emulator has a built-in Tandberg TDV-2215 emulator which is activated by setting terminal type to 52 or starting PED which will ask for the terminal type:

```
TERMINAL TYPES ARE:
 2:TELETYPE-ASR-33
11:DEC-LA36 (DECWRITER-II)
52:TANDBERG-TDV2215-SDS-V2
57:FACIT-4420-ND NOTIS
83:TANDBERG-TDV2200-ND NOTIS II
91:FACIT-TWIST (24-LINES MODE)
93:TANDBERG TDV2200/9S ND-NOTIS
103:Tandberg TDV2200/9S25 ND-NOTIS
 3:TANDBERG-TDV2115
36:TANDBERG-TDV2215-EX
53:TANDBERG-TDV2200/9-
80:TANDBERG TDV2200/9
90:TANDBERG TDV2200/9S
92:FACIT-TWIST (72-LIN
99:COLORTREND 210 ND
TERMINAL TYPE? 52_
```

```
PED:_- ND Program Editor - Version J - 1983-06-30
Line: 1-21 Column: 1-80 Region: MAIN
(..T::T..T..T.:T..T2.T..T..T.3T..T.:T..T..T.:T..T5.T..T..T
```

The emulator will show status lights at the bottom of the screen.
The font size etc. can be changed:



2.3 Mandatory to know

Read this section before you try to find SINTRAN-III functions that are not emulated.

2.3.1 BG-programs and RT-programs

The emulator is running background programs only, RT programs and most of the monitor calls associated with RT is not implemented. The RT-loader does not exist in the emulator and you can not look at segments.

2.3.2 The file system

SINTRAN-III is using file names coded in upper case and so does the emulator. You may however type lower case file names, they will be converted to upper case.

Most of the monitor calls for doing file-IO is emulated, some monitor calls that reads directory entries etc. are not emulated. You may open files for block access or sequential access and create new files. Direct access is not handled as in SINTRAN-III, but the user should not notice any difference.

The monitor calls for setting block size, reading no. of bytes in file etc. are implemented.

The emulator handles different users, i.e. file names can be (USER)FILE:TYPE and the open monitor calls cope with abbreviated user, file and type as in SINTRAN-III.

Directory names, however, are not emulated. Any attempt to use a (DIR:USER..... notation will cause problems for the emulator program

Creating new files using "FILE" (or "file") works.

The SINTRAN-III file version concept is not emulated, do not use "; " in the file names!

The file extension separator is "." , not ":" (which is illegal in Windows file names). The ND applications should still use : as file separator, the emulator will automatically substitute it with "." when required.

2.3.3 File indexing "NO SUCH PAGE"

In SINTRAN-III it is possible to have files with "holes", that is: You may write page 1 and page 99 of a file for example and access these randomly. The emulator does not emulate the SINTRAN-III file system with file indexing, it is using the native Windows/Linux file system which can not handle "holes". When writing e.g. page 99 of a file in the emulator the file will get 99*1024*2 bytes and afterwards it will be legal to access e.g. page 10, even if it has not really been written by the emulator. As long as your program does not use the "NO SUCH PAGE" error return to make decisions in the program logic, you will not have any problems with this nonconformance with SINTRAN-III. If you try to read beyond the maximum number of bytes, you will get an error return "NO SUCH PAGE" as in SINTRAN.

The “hole” syndrome also makes problems when transferring files from a ND-computer to Windows or Linux using ftp. The transfer will terminate because of the NO SUCH PAGE error before all of the file contents is transferred. This is the case for many :DATA files, but also for :PROG files for 2-bank programs.

A program that fills the holes is available and must be run on a ND-computer.

Note: When making :PROG files in the emulator, e.g. using BRF-LINKER, the program files will be created without holes and need not be fixed to be usable in the emulator later on.

2.3.4 Terminal break strategy

In SINTRAN-III you can decide whether your program is restarted or not for each byte typed on the terminal using the BREAK monitor call. This monitor call is also emulated, but the emulator restarts the user program for each byte independent of the break mode. The user program should see no difference. (The reason for implementing the BREAK monitor call in SINTRAN-III was to reduce the context switching when a large number of user worked simultaneously on the same computer. This should be no problem on a GHz computer with one user!) As a consequence of this strategy, the emulator does not buffer input characters from the keyboard, the ISIZE monitor call will therefore always return 0 bytes.

2.3.5 Parity

Many of the ND applications are generating and testing parity on text files. The QED editor is a typical example, you can control parity generation using the MPI(x) and MPO(x) modes. If you want to use e.g. Notepad to edit program files and compile then using e.g. FTN you will get a problem with “PARITY ERROR”.

You may work around this problem by using the command:

```
set-parity-when-reading-SYMB-files <on/off>
```

You may now use Notepad to edit files, save and then use it in e.g. MAC, NPL, FTN etc.

The emulator has a feature that will remove parity on output from an emulated ND program. Ref the command 3.44 use-7bit-on-symb-files-write <on/off>
If you activate this function before starting the ND program, output files will be readable by e.g. Notepad afterwards.

The emulator also has a command “remove-parity” that will remove on an existing file.

2.3.6 Users

As mentioned you may have different users. SYSTEM, RT and SCRATCH are mandatory. The user areas are simply folders named (SYSTEM), (RT) and (SCRATCH) and there are no restrictions on the number of pages available.

As in SINTRAN the emulator will search for files in SYSTEM's user area if not found at the logged in user. Only SYSTEM may write it's own files, others may read only. Except for user SYSTEM, there are no restrictions for reading and writing files between the users, friends are not emulated.

2.3.7 2-bank programs

The emulator supports 2-bank programs by emulating the Alternative Page Table mechanism. The ALTON and ALTOFF monitor is emulated and bit 0 in the status register may be turned on/off to control which bank to read/write from/to when in ALTON mode. There is no difference in execution time running in 1-bank or 2-bank mode.

2.3.8 Reentrant subsystems

This feature is not supported yet. The PLANC compiler seems to run, but for some commands the REENT monitor call is used and it will stop.

2.3.9 Floating point

The emulator handles both 32 and 48 bit floating point format. (48 bit is currently not working). The emulation is carried out by converting forth and back between the IEEE floating point format used by the ND computers and the IEEE format used by the Intel processors. Add, subtract, divide and multiply are therefore done by HW in the Intel processor. This may lead to some differences in the result of arithmetic operations between a ND computer and the emulator.

The instructions DNZ and NLZ is using a scaling factor that gives a one-to-one result, i.e. 3.000 is converted to 3 and nothing else!

2.3.10 Misc

Spooling files are not implemented.

2.3.11 Instructions not emulated

The instructions that are used for BCD is under development.
Stack instructions are currently not emulated.

3 The emulator command interface

Using the emulator interface which is command driven should not be a problem for a user known to SINTRAN-III. As in SINTRAN you may abbreviate the command until ambiguous and you will be asked for missing parameters. Some commands are intended to be identical to SINTRAN and can be executed from an application program through the MON 70 monitor calls. Such commands are preceded by an @ and can be listed by typing help @

Please note that the command interpreter is case sensitive (except for the SINTRAN commands).

If the command interpreter does not recognize a command it will try to locate an application program (.PROG file) at the logged in user's area. If it does not find a match there, it will look in user SYSTEM's area before giving up.

To start QED you only need to type QE , the Fortran compiler can be started by typing FO-100 etc. (This of course is provided that you have not made any files matching the name in your own user area)

If name a file that by some reason matches a command in the emulator, it can be started by typing !name.

Some of the commands have been introduced to be used for developing and debugging the emulator program and may not be very useful for the "normal" user.

3.1 upper-case <on/off>

Using this command you can control whether the characters typed on the keyboard are converted to upper case or not before returned to the calling application using the MON 1 monitor call.

It can be useful if you are working with e.g. mac which does not accept lower case letters. In practice this command works like the Caps Lock function on the keyboard.

3.2 log-in-user <user name>

When entering the emulator you are logged in as user EMULATOR.by default. Use this command to change user.

3.3 list-users

This command will list all known users in the DOS-window.

```
<>li-users
```

These users are known:

```
SCRATCH  
EMULATOR  
RT  
SYSTEM  
<>
```

3.4 trace-execution <on/off>

This command turns on or the execution trace function in the emulator. In trace mode the emulator will print the values of all the registers for each instruction executed. It also indicates which type of instruction that is executed. It is possible to start tracing when reaching a point in the application and it is possible to define trace areas as well. Trace output is directed to the DOS window where the emulator is running unless opening a trace file.

The trace output may be written to a file which may be investigated e.g. by Word Pad, Notepad or any other text processor program.

Note 1: Turning trace off does not close an open trace file.

Note 2: Turning on trace will influence the execution time.

3.5 trace-monitor-calls <on/off>

Turn on or off the tracing of monitor calls. The emulator will report all monitor calls in the emulator window or on the trace file together with the values of all registers.

Note: The monitor calls MON 1, MON 2, MON 71 and MON 72 are not traced because they are frequently used and generate a lot of output.

3.6 trace-stack <address>

Not implemented.

3.7 begin-trace-execution-when-at <address>

The trace function will be activated (and stay active) when the P-register reached the instruction at the value indicated by <address> which should be an octal number between 0 and 177777.

3.8 define-trace-area <trace no.> <lower> <upper>

Defines an area where the trace is active. Several trace areas may be active simultaneously. The upper and lower parameters should be octal numbers.

Tracing is not turned on by this command, you will have to activate tracing by the trace-execution command.

The trace areas that are defined can be listed by the "status" command.

3.9 reset-trace-area <trace number>

Deactivates a defined trace area.

3.10 open-trace-file <output file>

Use this command to open the file where the trace output will be written. Writing trace information is enabled via the trace-execution command. The file remains open until explicitly closed or when terminating the emulator program.

To create a new file embrace the file name by "xxxx".

The default file extension type is .txt and the file will be created in the ND110 folder, not in the logged in user area. To put it elsewhere use standard ../ notation, the command interpreter does not use SINTRAN-III notation.

3.11 close-trace-file

Closes the trace output file. The trace output will now appear in the command interpreter's DOS window.

3.12 set-break-point <address>

Set breakpoint at <address> which should be an octal number between 0 and 177777.

The emulator will stop execution when reaching a breakpoint.

Only one breakpoint can be defined.

To continue after a breakpoint use the command `continue-after-breakpoint`.

3.13 `clear-breakpoint`

Clear a defined breakpoint.

3.14 `continue-after-breakpoint`

Continue execution after having stopped at a breakpoint.

3.15 `define-histogram <start-address> <increment>`

Defines a histogram:

```
<>def-hist
Start address (octal): 300
Increment (octal): 1000
Histogram area starts at      300, ends at      12300
<>_
```

3.16 `list-histogram [optional output file]`

Running QED and exit at once makes this histogram:

```
<>li-hist
      Area      hits (dec)
-----
   300 -      1277         0
  1300 -      2277         0
  2300 -      3277       1567
  3300 -      4277       2889
  4300 -      5277         83
  5300 -      6277         0
  6300 -      7277         24
  7300 -     10277        263
 10300 -     11277         0
 11300 -     12277         0

Number of instr. executed: 5381
Inside histogram area      : 4826
Histogram has been cleared!
<>
```

3.17 `guard <location>`

Using this command the emulator will report whenever the contents of the memory address at `<location>` changes value. `<location>` should be an octal number: You may have up to 20 guards active simultaneously.

Guards are working in page table 0 only.

Guards cannot be reset but are cleared when a program is loaded into memory.

3.18 report-memory-reference <location>

Using this command the emulator will report whenever the contents of the memory address at <location> changes value. <location> should be an octal number between 0 and 3177777. Only one report can be active. (works in all page tables)

3.19 reset-report-memory-reference

Deactivates the defined report of a memory reference.

3.20 look-at-memory <page table> <address>

This command is similar to the SINTRAN-III LOOK-AT-MEMORY command. The page table should be 0 or 2 normally which is always used for BG-programs. The address should be a number between 0 and 177777 octal. (negative numbers are not accepted).

Example:

```
<>lo-im qed
```

```
Program low = 0 high = 20435 start = 0 restart = 1
Loaded OK, memory limits (bank 0) are now set to 0 - 20435
JMPi 125002 @ PC=      0 T=      0 A=      0 D=      0 X=      0 L=      0
ST=      0 B=      0
<>loo 0 0
```

```
Enter value (octal) or
"text string" or
memory address (octal value): (or /) to set location
Enter q or . and return to terminate
```

```
      0/      0 :      12  0/ 12 cntlch / cntlch >
0/      1 :      60542 141/142 ASCII a / ASCII b >
0/      2 :      61544 143/144 ASCII c / ASCII d >
0/      3 :      62506 145/106 ASCII e / ASCII F >
0/      4 :      43400 107/  0 ASCII G / cntlch >
0/      5 :          0  0/  0 cntlch / cntlch >
0/      6 :          0  0/  0 cntlch / cntlch >
```

Typing return advances to the next address.

Typing a value (octal) deposits the value (16 bits) in the current location and advances to next location.

Typing a value (octal) followed by / sets current location to the value.

Typing a . or q terminates examination mode.

Note that when 177777 is reached, the next location examined will be l the next page table.

Enter a character string by typing "abcde", (A termination char ' is not added)

3.21 find-value-in-memory <value> <page table>

List all the address of values that match <value> in the specified page table. <value> should be an octal number.

3.22 find-nonzero-in-page-table <page table no>

Find and list the memory addresses of all addresses that contains non-zero values.
(Used for development purposes)

3.23 fill-memory <value> <from> <to>

Fill the specified memory area with the given value. All numbers should be octal. A simple performance test can be done by fill-mem 0 0 177777 and then typing go 0. After 5 seconds type ESC and the number of instructions and the time pr. instruction is printed.

3.24 copy-memory <from> <bytecount> <ntimes>

Copies <bytecount> bytes starting at memory position <from> the number of times specified by <ntimes>. The memory position should be an octal number.

Note: Works in page table 0 only. (?)

3.25 set-register-value <register> <value>

Set the specified ND CPU register to <value> which should be an octal number.
Legal registers are T A D X B L S P (lower case acceptable).

If <register> equals * then all registers are set to <value>

Example:

```
<>set-reg * 0
```

```
(Bits in status register not influenced when setting other registers)
Status before:
LDX    54011 @ PC= 20145 T=    15 A=    0 D=    1 X=    55 L=   7370
ST=    142 B=    276
Status now:
LDX    54011 @ PC= 20145 T=    0 A=    0 D=    0 X=    0 L=    0
ST=     0 B=     0
<>
```

3.26 list-register-values

List the values of all registers.

Example:

```
<>li-reg
```

```
LDX    54011 @ PC= 20145 T=    15 A=    0 D=    1 X=   161 L=   7370
ST=    140 B=    276
<>
```

3.27 disassemble-memory <output file> <from addr> <to addr>

Not implemented.

3.28 status

Lists information about register values, active trace areas etc.

Example:

```
<>stat
```

```
CPU status is: 32 bit ND float format, defined program memory limits
are 0 - 20435
LDX 54011 @ PC= 20145 T= 15 A= 0 D= 1 X= 161 L= 7370
ST= 140 B= 276
Status bits: K= 0 Z= 0 Q= 0 O= 1 C= 1 M= 0 SSTG= 0 SSPTM= 0
Tracing memory references of address 123
Trace area number 1: 100 - 1777
<>
```

3.29 dummy

Has no function.

3.30 ! <image file>

Use this command to start a ND program file in the emulator if the name of the program file you want to start is identical to a command name. If you for example make a program file with the name HELP.PROG, the command interpreter would recognize "help" as the command that lists matching commands of the emulator.

Typing HELP would start the program since the command interpreter is case sensitive. (The ! feature was introduced at a stage in the development of the emulator when prog-files also was case sensitive)

Example:

```
<>! qed
```

```
Program low = 0 high = 20435 start = 0 restart = 1
Loaded OK, memory limits (bank 0) are now set to 0 - 20435
JMPi 125002 @ PC= 0 T= 15 A= 0 D= 1 X= 161 L= 7370
ST= 140 B= 276
Starting execution at start address 0
```

QED 4.3

*

If the file to execute is not found in the logged in user's area, user SYSTEM will be investigated to see if the file exist there.

Explicitly the user name may be specified:

```
<>! (SYSTEM)NRL
```

```
Program low = 0 high = 24631 start
Loaded OK, memory limits (bank 0) a
JMPi 125002 @ PC= 0 T= 15 A
Starting execution at start address
```

```
RELOCATING LOADER LDR-1935J
```

```
*
```

Note that the program file name is not case sensitive.

3.31 load-image-file <image file>

This command loads a program file into memory, but does not start execution. The parameter <image file> specifies a ND executable :PROG file.

If the file to execute is not found in the logged in user's area, user SYSTEM will be investigated to see if the file exist there.

Explicitly the user name may be specified:

Example:

```
Segment tables cleared
```

```
Open file: ./(SYSTEM)/QED.PROG
```

```
Program low = 0 high = 20435 start = 0 restart = 1
```

```
Loaded OK, memory limits (bank 0) are now set to 0 - 20435
```

```
JMPi 125002 @ PC= 0 T= 0 A= 0 D= 0 X= 0 L=
```

```
0 ST= 0 B= 0
```

```
Check if program loaded on overlay segment: NO
```

```
<>
```

3.32 load-byte-swapped-image-file <image file>

This command is identical to the "load-image" command, except for that it expects the image file to be "byte swapped", that is the byte ordering of the ND file has not been fixed to match the Intel architecture. This command can be used if you by some reason have problems with transferring binary files from the a ND computer.

3.33 set-memory-limits <lower> <upper>

Use this command to set the memory limits of the program area. The values of the lower and upper parameters should be octal numbers between 0 and 177777.

The memory limits are used when dumping memory to file, but only when in 1-bank mode.

3.34 set-2bank-mode <on/off>.

Set the emulator in 2-bank or 1-bank mode. The bank mode only influences how the "dump-memory" command will create the ND :PROG file.

Setting 2-bank mode is relevant only when you e.g. make a 2-bank program using MON ALTON in NPL or MAC and dump it. Making program files using the BRF-LINKER is working as it does on a ND computer.

Some times a program file that is transferred from a ND computer is recognized by the emulator as a 2-bank program even if it is a 1-bank program. In these cases the program will be executable without any problems, but an error messages is printed each time it is invoked.

To get rid of the error message do the following:

```
load-image xxx
set-2b off
dump xxx sa ra
```

3.35 activate-alternative-page-table <APT no.>

This command turns on alternative page table (like MON ALTON). It can be used to e.g. test small programs to see how the alternative page table mechanism works. This command was implemented to test the APT mechanism in the emulator.

3.36 save-image-to-file <image file> <SA> <RA>

Identical to “dump-memory”

3.37 dump-memory <prog.file> <start address> <restart address>

Dumps the emulator’s memory to an executable ND file with start and restart addresses that will be used when invoking the program using the ! command. Start and restart address are octal numbers. To create a new ND :prog file, embed the file name in “”.

If the emulator is in 1-bank mode the area in page table 0 defined by the “set-memory-limits” determines the area to be dumped to file.

If the emulator is in 2-bank mode the entire contents of page table 0 and 2 will be dumped to the :PROG file. This is done to cope with the problem of files with “holes”, ref . 2.3.3.

Switching the emulator between 1-bank and 2-bank mode is done using the command “set-2bank-mode <on/off>”.

Using 2-bank mode is only sesible when you make a program that turns on the alternative page table by means of MON ALTON.

2-bank files always use 256 kbytes file space.

3.38 dummy2

Has no function.

3.39 run <start-address>

Start the emulation at the given address which must be an octal number between 0 and 177777.

Identical to “@go”.

3.40 continue-at-restart-address

Continue execution at the restart address (RA) of the currently loaded ND application program.

3.41 help <command> <output file>

List all commands matching <command> to the specified output file. If no outfile is defined, the emulator DOS window will be used for output. To create a new file, embed the file name with "". You may specify a directory path ../ , the emulator is not using SINTRAN-III user notation for this command.

Example:

```
<>help load,,,,  
  
load-image-file <image file>  
load-byte-swapped-image-file <image file>  
<>
```

3.42 open-scratch-file

The scratch file (SCRATCH01:DATA) is opened when starting the emulator. If it is closed by some reason it may be reopened using this command.

3.43 check-files

Checks the files present in the logged in user's area to find files that are accessible as SINTRAN-III files. The example below shows the list of files that are not accepted and which are accepted. Files that are not accepted may be present, but they may not be opened by ND programs.

```
<>log-in sys
```

Logged out user EMULATOR and logged in user SYSTEM (user 0)

```
<>check
```

```
File:                                FFF-SYMB > File type is missing  
File:          A2345678901234567.SYMB > File name is too long  
File:                                LLL.SYMBX > File type is too long
```

These files are known for user SYSTEM:

```
BRF-EDITOR.PROG  
FTNLIBR.BRF  
ABC.SYMB  
BRF-LINKER.PROG  
DDBTABLES-C.VTM  
FILE-EXTRACT.PROG  
FMAC.PROG  
FORTRAN-100-G02.PROG  
FORTRAN-1B-G02.BRF  
FORTRAN-2B-G02.BRF  
FTN.PROG  
LIST-ALL-FILES.PROG
```

```
LOOK-FILE . PROG
MAC . PROG
NPL . PROG
NRL . PROG
OOO . SYMB
PED . PROG
PED-ENG-J . HELP
QED . PROG
UE-ERMSG-EN-B . ERR
UE-ERRORS-ENG-A . ERR
LLL . SYMB
KKK . SYMB
<>
```

3.44 use-7bit-on-symb-files-write <on/off>

If 7 bit mode is turned on the emulator will remove the parity bit for each byte when writing to a :SYMB file (works for seq. and random access). This can be convenient if you want to read a file generated by a ND program by e.g. Notepad or Wordpad.

3.45 float <format>

Sets the emulator to emulate 32 bit or 48 bits floating point. 32 bit floating point is default. Format is either 32 or 48.

*** 48 bit float is currently not implemented ***

3.46 compare-images <image file> <image file> [report file]

Compares two ND program files (:PROG) and writes a report of differences in SA, RA, memory limits and the dumped memory area. [report file] is optional. It is possible to compare both 1-bank and 2-bank program files.

The files names are not using SINTRAN-III user names etc., directory paths ../../ may be specified. File name abbreviation is not possible.

3.47 add-parity <input file> <output file>

Reads the input file byte by byte, generates even parity and writes the result to the output file.

The files names are not using SINTRAN-III user names etc., directory paths ../../ may be specified. File name abbreviation is not possible.

3.48 remove-parity <input file> <output file>

Reads the input file byte by byte, clears bit 7 and writes the result to the output file.

The files names are not using SINTRAN-III user names etc., directory paths ../../ may be specified. File name abbreviation is not possible.

3.49 byte-swap <inp.file> <outp.file>

Reads the input file two bytes at the time from the input file and writes them in the opposite order to the output file.

3.50 od <format> <input nfile>

Generates an octal byte (8bits) or word (16 bits) dump of the specified input file. The default file name is :BIN. Format is either 8 or 16.

Example:

```
<>od 8 ff
Dump from file ./ (EMULATOR) /FF.BIN
  0 (    0):    0  1  0  2  0  3  0  4  0  5  .....
 10 (   12):    0  6  0  7  0 10  0  0  0  0  .....
 20 (   24):    0  0  0  0  0  0  0  0  0  0  .....
 30 (   36):    0  0  0  0  0  0  0  0  0  0  .....
```

Generates an octal word (16 bits) dump of the specified file. The default file name is :BIN.

Example:

```
<>od 16 ff
Dump from file ./ (EMULATOR) /FF.BIN
<>
   0 (    0):          1          2          3          4          5
   5 (    5):          6          7         10          0          0
  10 (   12):          0          0          0          0          0
  15 (   17):          0          0          0          0          0
```

3.51 octal-to-decimal <number>

Prints the decimal value of the octal <number>.

3.52 decimal-to-octal <number>

Prints the octal value of the decimal <number>.

3.53 float-test <val1> <val2>

Prints the floating point value represented by the two octal numbers as if value1 resided in the A-register and value 2 in the D-register.

Example:

```
<>flo-test 40000 0

Converted to 0.500000

Convert back to ND format: 40000 , 0

<>
```

Note: 32 bit floating point format only.

3.54 mode <input file> <output file>

NOTE: When directing output to the <output> file the emulator commands and output will still be printed in the console window not on the <output file>. Needs to be fixed.

Using this command is similar to the @MODE command in SINTRAN-III, typically used to compile and link programs etc. The mode file should use lower case letters for the emulator commands which should not be preceded by an @ as in SINTRAN-III mode files.

The emulated SINTRAN-III commands must be preceded by an @, e.g. @li-fi
This command use SINTRAN-III abbreviation of the file name and a user name may be specified, the default file extension is :MODE
You may use e.g. Notepad to generate the mode file.

Example: (file list depending on what is in the user area)

Contents of mode file:

```
@li-fi,,,
npl
@FLO48
@DEV SIMULATOR-1,,100
MAC
) 9ASSM 100
) 9END
) 9TSS
set-mem 0 177777
dump "SIM" 0 1
```

```
<>mode comp,,,
```

```
<>@li-fi,,,
```

```
FILE 0 : (EMULATOR) AIM:6502
FILE 1 : (EMULATOR) AIM:PROG
FILE 2 : (EMULATOR) BCD:SYMB
FILE 9 : (EMULATOR) COMP-AIM:MODE
FILE 10 : (EMULATOR) COPFTEST:SYMB
FILE 11 : (EMULATOR) D:PROG
FILE 12 : (EMULATOR) D2TEST:PROG
FILE 13 : (EMULATOR) DO:PROG
FILE 19 : (EMULATOR) GETRT:SYMB
FILE 20 : (EMULATOR) GETRTDESC:SYMB
FILE 31 : (EMULATOR) SIMULATOR-1:SYMB
FILE 32 : (EMULATOR) SIMULATOR-2:SYMB
FILE 36 : (EMULATOR) SUB1:SYMB
FILE 37 : (EMULATOR) TEST-BFILL:SYMB
FILE 38 : (EMULATOR) TEST-BYTEMOVE:PROG
FILE 39 : (EMULATOR) TEST-BYTEMOVE:SYMB
FILE 40 : (EMULATOR) TEST-FORT:MODE
FILE 41 : (EMULATOR) TEST-FORTTRAN:MODE
FILE 42 : (EMULATOR) TEST-FTN:MODE
```



```
FILE 54 : (EMULATOR) ZZZ:SYMB
<>npl
Open file: ./ (SYSTEM) /NPL.PROG

NORD PL NOVEMBER 1979

@FLO48
@DEV SIMULATOR-1,,100
  WARNING: SUBROUTINE ENTRYPOINT CHANGED TO ,SA

- END OF COMPILATION
0 ERRORS DETECTED
Time used is 0 secs, 654 milsecs
77080764 instructions executed, 0.008498 usec/instruction
<>
<>MAC
Open file: ./ (SYSTEM) /MAC.PROG

- MAC -
)9ASSM 100
  **** 000000 DIAGNOSTICS ****

)9END
)9TSS
Time used is 0 secs, 381 milsecs
45645257 instructions executed, 0.008369 usec/instruction
<>
<>set-mem 0 177777

Memory limits are now      0 - 177777
<>dump AIM 0 1
Open file: ./ (EMULATOR) /AIM.PROG
Successfully saved program code to file

<>
End of file on mode input file
<>
```

3.55 exit

Terminates the emulator program.

3.56 quit

Terminates the emulator program.

3.57 debug <level(0=off)>

If set to a nonzero (positive) value the emulator will print some debug information, mostly from the emulation of the monitor calls.

Used for emulator debugging purposes.

3.58 zz

Do not use this command!

3.59 assemble-to-memory <start address>

Not implemented.

3.60 READ-BINARY <input file>

Not implemented.

3.61 WRITE-BINARY <page table> <output file>

Write the memory content in the specified page table to the specified file. The default file type is :BIN

Page table is either 0,1,2 or 3. Page table 0 contains program code and data if running 1-bank programs and program code if running 2-bank programs. When running 2-bank programs data will be found in page table 2.

The entire memory is dumped. (64K 16 bit words).

In SINTRAN page table 1 is used for the OS. In the emulator page table 0 is used for user programs.

3.62 define-segment <segment number > <name>

This command is used to associate a segment name with a segment number.

An alternative way to define segments/names is to use a text editor, e.g. Notepad or run PED in the ND emulator. You may add, change or delete segments using a text editor.

NOTE: If using QED type MPI(0) before you read the file and type MPO(0) before you write back to turn off the parity bit check when reading and parity bit generation when writing.

3.63 load-segment <segment number> <progfile>

This command creates a file on user SYSTEM and requires that a segment has been defined on beforehand. The name of the file will be like this:

 SEGFILE_200.SGNO	25.09.2017 10.59	SGNO-fil	129 kB
 SEGFILE_201.SGNO	25.09.2017 11.00	SGNO-fil	133 kB
 SEGFILE_202.SGNO	25.09.2017 11.00	SGNO-fil	103 kB
 SEGFILE_203.SGNO	25.09.2017 21.19	SGNO-fil	28 kB
 SEGFILE_204.SGNO	25.09.2017 11.00	SGNO-fil	141 kB

The name is SEGFILE_xxx. SGNO where xxx is the segment number.

The second parameter <progfile> must be a program file, :PROG, linked as a 2-bank program. Refer to the ND manual:

ND-60.196.3 BRF-LINKER

<http://sintran.com/sintran/library/libsw/ND-60196-3-EN.pdf>

3.64 list-segments

This command will list the defined segments from the segment file.

./(SYSTEM)/SEGFILE.SIN3

Example:

```
<>li-seg
Segment no-name-progfile name: 200 - WPEDITM - wp-editor-m06
Segment no-name-progfile name: 201 - WPCMDM - wp-cmd-m06
Segment no-name-progfile name: 202 - WPUTILM - wp-util-m06
Segment no-name-progfile name: 203 - WPMMAIN - wp-main-no-m06
Segment no-name-progfile name: 204 - WPPRMM - wp-prm-m06
<> _
```

3.65 delete-segment

Not used, use a text editor instead and edit the contents of the (SYSTEM)/SEGFILE.SIN3.

3.66 list-SINTRAN-commands

List the commands that are implemented to be SINTRAN-III identical (at least close to).

3.67 SINTRAN-III commands description

3.67.1 @set-terminal-type <n>

Use this command to set the terminal type.

PED is dependant of recognizing a terminal type. (PED will show a list of recognized terminal types, chose 52)

At start the emulator sets terminal type 0.

3.67.2 @list-open-files

Lists the files that are currently open. When issued as a command from the emulator command interpreter only the scratch file should be open.

```
<>@li-op
```

```
100: ./(SCRATCH)/SCRATCH01.DATA Opened for access 2, block size
512 bytes
```

```
<>
```

When issued from a ND program all files currently open will be listed.

```
PED:@li-op
```

```
100: ./(SCRATCH)/SCRATCH01.DATA Opened for access 2, block size
2048 bytes
```

```
101: ./(SYSTEM)/UE-ERRORS-ENG-A.ERR Opened for access 2, block
size 2048 bytes
```

PED:

The format is somewhat different from SINTRAN-III output.

3.67.3 @close-file <file number (-1=all)>

Closes a specified file where file number typically is 100, 101 ... (octal).
If file number is -1 all open files except the scratch-file will be closed.

3.67.4 @list-file <file name>

List all files matching <file name> in the DOS window.

```
<>@li-fi (sys) fo,,,
```

```
FILE 7 : (SYSTEM) FORTRAN-100-G02:PROG
FILE 8 : (SYSTEM) FORTRAN-1B-G02:BRF
FILE 9 : (SYSTEM) FORTRAN-2B-G02:BRF
<>@li-fi (sys):brf,,,
```

```
FILE 1 : (SYSTEM) BASLIBR-H00:BRF
FILE 11 : (SYSTEM) FORTRAN-1BANK:BRF
FILE 12 : (SYSTEM) FORTRAN-2BANK:BRF
FILE 14 : (SYSTEM) FTNLIBR:BRF
FILE 15 : (SYSTEM) FTNRTLBR-2094F:BRF
FILE 17 : (SYSTEM) JEC-1BANK-B00:BRF
FILE 18 : (SYSTEM) JEC-2BANK-B00:BRF
FILE 23 : (SYSTEM) MON-CALL-1B-A00:BRF
FILE 24 : (SYSTEM) MON-CALL-2B-A00:BRF
FILE 36 : (SYSTEM) PLANC-1BANK-I01:BRF
FILE 37 : (SYSTEM) PLANC-2BANK-I01:BRF
FILE 38 : (SYSTEM) PLANC-UTILLIB-1B:BRF
FILE 39 : (SYSTEM) PLANC-UTILLIB-2B:BRF
<>
```

3.67.5 @new <user name>

Works similar to the NEW command in SINTRAN-III, except for that user RT also may switch to any user as SYSTEM can. When starting the emulator user EMULATOR is logged in. To change to user SYSTEM use the command log-in-user.

3.67.6 @who

Prints the current user:

```
<>@who
```

```
==> 1 EMULATOR
<>
```


146145
004016
134006
155570
004373
134003
060371
125010
170404
164403
164402
175235
124376
164400
146142
000000
125001
000033
177777!
Found end of preamble (!)
Load start address is 33
Data word count is 11
Checksum is 11002
Action is 0
◁run 33
@@
◁

3.67.8 @place-image-file <.PROG file>

Loads a PROG-file image into memory without starting execution.

◁place qed

Segment tables cleared
Open file: ./(SYSTEM)/QED.PROG

Program low = 0 high = 20435 start = 0 restart = 1
Loaded OK, memory limits (bank 0) are now set to 0 - 20435
JMPi 125002 @ PC= 0 T= 0 A= 0 D= 0 X= 0 L= 0 ST= 0 B= 0
Check if program loaded on overlay segment: NO

3.67.9 @go <address>

Start the emulation at the given address which must be an octal number between 0 and 177777.
Identical to "run".

3.67.10 @recover <prog file>

Loads and starts a program file, default file type is PROG. The emulator will search for the program file in the logged in user file folder. If it is not found, it will search for the file in user (SYSTEM).

A simpler way of starting a program is to simply type the file name at the command prompt, e.g. <>qed instead of <>@recover qed. If you have named the program file such that it matches an emulator command the command will have priority.

OR: You may type <>! qed

3.67.11 @delete-file <file name>

Identical to the SINTRAN-III command. The file name may be abbreviated, ambiguous file name will be detected. The default file type is SYMB.

@DELETE-FILE TEST-1	Deletes the file TEST-1.SYMB of current user
@DELETE-FILE (user)TEST-1	Deletes the file TEST-1.SYMB of (user)
@DELETE-FILE TEST-1:BRF	Deletes the file TEST-1.BRF of current user

The file to delete must be upper-case in the file folder, but may be lower case in the command. (it will be converted to upper case.)

3.67.12 @create-file <file name>

Creates a file in the logged-in user folder. The default file type is SYMB.
(this command needs further development!)

3.67.13 @copy-file <to file> <from file>

Copies a file to another. Default file type is SYMB.

Copies from file test to file xxx which is created:

```
<>@COP-FILE "eee" COPF
Create file ./(EMULATOR)/EEE.SYMB
Open file: ./(EMULATOR)/EEE.SYMB
Open file: ./(EMULATOR)/COPFTEST.SYMB
21 bytes copied
<>
```

File type is BRF, destination file exists:

```
<>@COP-FILE sub1:brf sub1-xx:brf
Open file: ./(EMULATOR)/SUB1.BRF
Open file: ./(EMULATOR)/SUB1-XX.BRF
193 bytes copied
<>
```

3.68 SINTRAN-info

Lists the information about the SINTRAN III operating system and the CPU. The data is fetched from a real ND computer. The data can be retrieved from a user program by the monitor call GetSystemInfo (MON 262).

The command <>float <format> will change the value for the CPU type . (4 or 5).

The PLANC compiler is using this information..

```
<>SINTRAN
System number (CPU #): 16505
CPU type              : 5
Instruction set        : 10
Microprogram version  : 12
System type           : 0
Operating system       : 5
Operating system vers: 0
Patch level           : 64
Sys.gen time          : 09:34 16/12-1988
```

3.69 find-in-file <file> <w1> <w2>

This command can be used to find two consecutive 16 bit numbers in a file.

The numbers should be entered as octal numbers.

3.70 BCD-test <value 1> <operation> <value 2>

This command adds/subtracts/multiplies/divides two ASCII BCD numbers and is used for test purpose.

```
BCD-test <value 1> <operation> <value 2>
<>BCD 1234.45 + 679
```

Result is : 1913.45 (No. of decimals: 2)

<>

3.71 test-input

This command can be used to check the keycodes of the keyboard

```
<>te-inp
```

Type ESC to exit

Key: 100(dec), 64(hex), 144(oct)

Key: 0(dec), 0(hex), 0(oct)

Key: 115(dec), 73(hex), 163(oct)

Key: 0(dec), 0(hex), 0(oct)

Key: 97(dec), 61(hex), 141(oct)

Key: 0(dec), 0(hex), 0(oct)

3.72 CPU-type <cpu type no.> <instruction set no.>

<>CPU 1 10

System number (CPU #): 16505
CPU type : 1
Instruction set : 10
Microprogram version : 12
System type : 0
Operating system : 5 SINTRAN III VSX-500
Operating system vers : D
Patch level : 64
Sys.gen time : 09:34 16/12-1988

<>

3.73 set-parity-when-reading-SYMB-files <on/off>

If you generate a source file using Notepad ND programs like QED, NPL, PED and MAC will report parity errors because they are testing each byte for even parity when reading the ASCII signs. This command will tell the emulator to generate ASCII code with even parity which is accepted by the programs.

3.74 check-monitor-call <number>

Prints a list of the monitor calls that are implemented in the emulator. Enter 555 to print the complete list.

3.75 silent-mode <on/off>

When turning silent mode off the emulator will write extra information when e.g. starting a program etc. At start-up silent mode is on.

3.76 more

Simply copies the contents of the specified file to the console window. Default file type is SYMB.

4 Stopping a running ND application

The ESC button will terminate the running application if ESC has not been disabled by the ND monitor call for this purpose, but only if the program comes to a MON 1 instruction. (In SINTRAN-III it will be stopped unconditionally)

To stop a program that goes into an endless loop, type ctrl-C on the keyboard. (ctrl-C will not terminate the application if it is currently in MON 1).

If unable to stop using any of these methods simply kill the DOS window.

5 Planned development

5.1 Reentrant subsystems

Emulate the monitor call that enables reentrant subsystems.

6 Monitor calls emulated

Use the command <>check-monitor-call <number> to get a list of the monitor calls that are implemented in the emulator

6.1 Nonconformance monitor calls

To be written....

7 List of emulator commands

upper-case <on/off>
close-trace-file
open-trace-file <output file>
trace-execution <on/off>
look-at-memory <page table> <address>
test-font-set
run <start-address>
set-memory-limits <lower> <upper>
save-image-to-file <image file> <SA> <RA>
copy-memory <from> <bytecount> <ntimes>
define-trace-area <trace no.> <lower> <upper>
reset-trace-area <trace number>
status
report-memory-change <address>
set-break-point <address>
clear-breakpoint
continue-after-breakpoint
@recover <prog file>
help <command> <output file>
fill-memory <value> <from> <to>
assemble-to-memory <start address>
quit
load-image-file <image file>
find-value-in-memory <value> <page table>
step-execution [optional start address]
define-histogram <start-address> <increment>
list-histogram [optional output file]
ZZ
reset-report-memory-reference
set-register-value <register> <value>
exit
list-register-values
disassemble-memory <output file> <from addr> <to addr>
float <32 or 48 bits>

trace-monitor-calls <on/off>
READ-BINARY <input file>
WRITE-BINARY <page table> <output file>
COMPARE-BINARY <file 1> <file 2>
trace-stack <address>
begin-trace-execution-when-at <address>
load-byte-swapped-image-file <image file>
guard <page table> <address>
use-7bit-on-symb-files-write <on/off>
mode-input-file <file name>
continue-at-restart-address
! <image file>
octal-to-decimal <number>
decimal-to-octal <number>
compare-images <image file1> <image file2> [report file]
add-parity <input file> <output file>
remove-parity <input file> <output file>
float-test <val1> <val2>
activate-alternative-page-table <APT no.>
find-nonzero-in-page-table <page table no>
@set-terminal-type <n>
set-2bank-mode <on/off> <page table>
go <address>
dump-memory <prog.file> <start address> <restart address>
od <8 or 16 bit> <input file>
byte-swap <inp.file> <outp.file>
@list-open-files
@close-file <file number (-1=all)>
open-scratch-file
debug <level(0=off)>
log-in-user <user name>
list-users
@list-file <file name>
check-files
@new <user name>
@who
list-segments
define-segment <no> <name>
delete-segment <no>
load-segment <segno> <prog-file>
clear-internal-segment-table
SINTRAN-info
find-in-file <file> <w1> <w2>
BCD-test <value 1> <operation> <value 2>
test-input
place-image-file <.PROG file>
@place-binary <.BPUN file>
CPU-type <cpu type no.> <instruction set no.>
set-parity-when-reading-SYMB-files <on/off>-table
check-monitor-call <number>

@delete-file <file name>
@copy-file<from file> <to file>
list-SINTRAN-commands
silent-mode <on/off>
@create-file <name>
more <file>

7.1 The PREP2B program

When transferring files from the ND computer e.g. using ftp an error will occur if the files has “holes”, i.e. missing pages. If a file for example has page 0,1,2 and 10 written, pages 3,4,5,6,7,8 and 9 is not accessible. This is the way the SINTRAN-III file system works, while Windows and Linux do not allow files with “holes” in them. The emulator does not emulate the SINTRAN-III file system, which means that it is actually possible to access a file written by an emulator program even if the page is not written. If you for example open a file and write page 10, you will be able to read pages 0-9 as well number 10.

To be able to transfer files using ftp from a ND computer it will need some preparation if it has missing pages. Typically 2-bank program files need to be prepared, other files like :DATA files can be candidates for preparation as well.

The program PREP2B reads a file and writes the contents to another file checking for empty pages in the input file and writing dummy data if found.

The program looks like this:

```
PROGRAM RFIL
INTEGER*2 IDATA(1024)
INTEGER*2 NPAGES
INTEGER*4 NBYTES
C  INTEGER ERRCODE
INTEGER*2 IFILE, OFILE
CHARACTER*80 INNFILE, OUTFILE
OUTPUT(1)'This program reads a :PROG file dumped in 2-bank mode'
OUTPUT(1)'and writes it to the specified output file while'
OUTPUT(1)'filling in the missing file pages that cause problems'
OUTPUT(1)'when trying to ftp these type of files (NO SUCH PAGE)'
OUTPUT(1)'Output file size will be 129 pages ~ 250 kbytes'

WRITE(1)6412B,'Enter input file name: '
INPUT(1)INNFILE
WRITE(1)6412B,'Enter output file name: '
INPUT(1)OUTFILE

IFILE = 7
OFILE = 8
OPEN(IFILE,STATUS='OLD',FILE=INNFILE, ACCESS = 'RX'
C , ERR=666)
CALL      RMAX(IFILE,NBYTES)
output(1)'Bytes in input file: ', NBYTES
OPEN(OFILE ,FILE=OUTFILE, ACCESS = 'WX', ERR= 777)
CALL SETBS(OFILE,1024)
CALL SETBS(IFILE,1024)
```

```
IF ( ERRCODE .NE. 0 ) THEN
    OUTPUT(1)'SETBS error'
ELSE
    NPAGES = (NBYTES+1)/2048
    NPAGES = NPAGES + 1
    OUTPUT(1)'Total pages to write: ',NPAGES
    DO FOR I = 1,NPAGES
        CALL RFILE(IFILE,0,IDATA,I-1,1024)
        IF ( ERRCODE .NE. 0 ) THEN
            OUTPUT(1)'No data found bl.no : ',I-1,
C            ' ..write data to fill missing page'
            DO FOR J = 1,1024
                IDATA(J) = 333
            ENDDO
            CALL WFILE(OFILE,0,IDATA,I-1,1024)
            IF ( ERRCODE .NE.0 ) THEN
                OUTPUT(1)'Write failed on bl.no: ',I-1
                OUTPUT(1)'Terminate! (check user space?)'
                CLOSE(IFILE)
                CLOSE(OFILE)
                GO TO 9999
            ENDIF
        ELSE
            OUTPUT(1)'Success on block: ',I-1
            CALL WFILE(OFILE,0,IDATA,I-1,1024)
            IF ( ERRCODE .NE.0 ) THEN
                OUTPUT(1)'Write failed on bl.no: ',I-1
                OUTPUT(1)'Terminate! (check user space?)'
                CLOSE(IFILE)
                CLOSE(OFILE)
                GO TO 9999
            ENDIF
        ENDIF
    ENDDO
ENDIF

CLOSE(IFILE)
GO TO 9999

666 OUTPUT(1)'OPEN input file', INNFILE,' failed'
GO TO 9999
777 OUTPUT(1)'OPEN output file', OUTFILE,' failed'
CLOSE(IFILE)
9999 OUTPUT(1)'PROGRAM END'
END
EOF
```

8 System devices

The files `TERMINAL.SYMB` and `LINE-PRINTER.SYMB` should reside on user `SYSTEM`.

TERMINAL

You may open `TERMINAL`, e.g. in mac

```
)9ASSM XXX,TERM,0
```

Which is identical to

)9ASSM XXX,TERM,0

The file number for TERMINAL is always 1.

LINE-PRINTER

Is currently not working, you will get an error message NOT WRITE ACCESS.
The intention is to print directly to a printer. Future extension.

9 Segment overlay

Segment overlay makes it possible to run program code that is larger than 64k words.
The maximum data space (bank 2) is 64k words.

The program code is placed on different segments and the emulator will switch between these segments automatically when required.

The commands

<>list-segments

<>load-segment

are important when generating segment overlay programs.

The file:

(SYSTEM)SEGFIL.SIN3

Contains the number, name and associated program names of all defined segments and is a readable file that can be edited e.g. by PED or QED or a text editor like Notepad.

The command <>list-segments will print the contents of the file:

```
<>li-seg

Segno / name / progfile name

200 / WPEDITM / wp-editor-m06
201 / WPCMDM / wp-cmd-m06
202 / WPUTILM / wp-util-m06
203 / WPMMAIN / wp-main-no-m06
204 / WPPRMM / wp-prm-m06
300 / T1 / segtest
302 / T2 / sub1
303 / T3 / test3
777 / TS / testtest
** Missing parameter in line 10
555 / FTY / kkkk
<>_
```

In the example below we will use the two segments 300 and 302

An example of how to generate an overlay program in Fortran.

This program is printing a text on the screen, then calling a subroutine residing on another program segment SUB1. This subroutine (SUB1) will print a message and return to the main program on segment MAIN. Before the main program terminates it will print a second message.

The main program code:

```
PROGRAM SIMPLE
WRITE (1,*) 'This is the main program'
CALL SUB1
WRITE (1,*) 'this is the main program again'
END
```

The subroutine

```
SUBROUTINE SUB1
WRITE (1,*) 'This is SUB1'
END
```

The files reside on the on user (EMULATOR) on SIMPLE-MAIN.SYMB and SUB1:SYMB.

Compiling using the FORTRAN-100 compiler

Type fort-100 at the emulator command line. When the compiler starts it will print

ND-100/NORD-10 ANSI 77 FORTRAN COMPILER - 210191G02
FTN:

Compiling the main program:

```
ND-100/NORD-10 ANSI 77 FORTRAN COMPILER - 210191G02
FTN: separate-data on
FTN: compile simple-main,,simple-main
- CPU TIME USED: 0.0 SECONDS. 5 LINES COMPILED.
- NO MESSAGES
- PROGRAM SIZE=46 DATA SIZE=77 COMMON SIZE=0
FTN: compile sub1,,sub1
- CPU TIME USED: 0.0 SECONDS. 3 LINES COMPILED.
- NO MESSAGES
- PROGRAM SIZE=19 DATA SIZE=29 COMMON SIZE=0
FTN: ex
```

Linking:

<>brf-linker

- BRF Linker - 10721B00

Br1: prog-file simple-main/T1

Br1: define #dclc 2000 d

Br1: load simple-main

FREE: P 000656-177777 D 002114-177777

Br1: load fort-2b

FORTTRAN-2BANK-G02 32-BIT FLOATING

PLANC-2BANK-H00

FREE: P 032301-177777 D 011307-177777

Br1: exi

SUB1.....32304 U

!! restart brf-linker to load the second program

<>brf-linker

Br1: pro-file sub1/T2

Br1: define #dclc 14000 d

Br1: link-to simple-main

Br1: load sub1

FREE: P 000623-177777 D 014034-177777

Br1: load fort-2b

FORTTRAN-2BANK-G02 32-BIT FLOATING

PLANC-2BANK-H00

FREE: P 032040-177777 D 023227-177777

Br1: exit

5PTAB....13511 U linked from SIMPLE-MAIN

5TR32AP..24310 U linked from SIMPLE-MAIN

5EXCINF..14033 U linked from SIMPLE-MAIN

5ESTACK..23334 U linked from SIMPLE-MAIN

5STACK....1053 U linked from SIMPLE-MAIN

5FIO_BL..14031 U linked from SIMPLE-MAIN

5USFILB..21070 U linked from SIMPLE-MAIN

5CNCT....21071 U linked from SIMPLE-MAIN

5ALTREC..24420 U linked from SIMPLE-MAIN

SUB1.....600 P linked to SIMPLE-MAIN

<>

Create segments if not existing:

Use PED and edit the file (SYSTEM)SEGFILE:SIN3

Load segments.

```
<>load-segment 300 simple-main
    The emulator will output some info....
<>load-segment 302 sub1
    The emulator will output some info....
```

For some reason the program will fail if starting it from user EMULATOR.
The SIMPLE-MAIN:PROG file must be copied to user SYSTEM and started from there.

```
<>simple-main
...
This is the main program
This is SUB1
*** 00-00-00 00:00:00 SIMULATED TRAP: (7633B)
STACK OVERFLOW
RETURN ADDRESS 725B

*** JOB ABORTED ***
```

```
<>@cop-file (sys)simple:p simple:p
Open file: ./(SYSTEM)/SIMPLE-MAIN.PROG
Open file: ./(EMULATOR)/SIMPLE-MAIN.PROG
395264 bytes copied
<>
```

```
<>(sys)simple
This is the main program
This is SUB1
this is the main program again
```